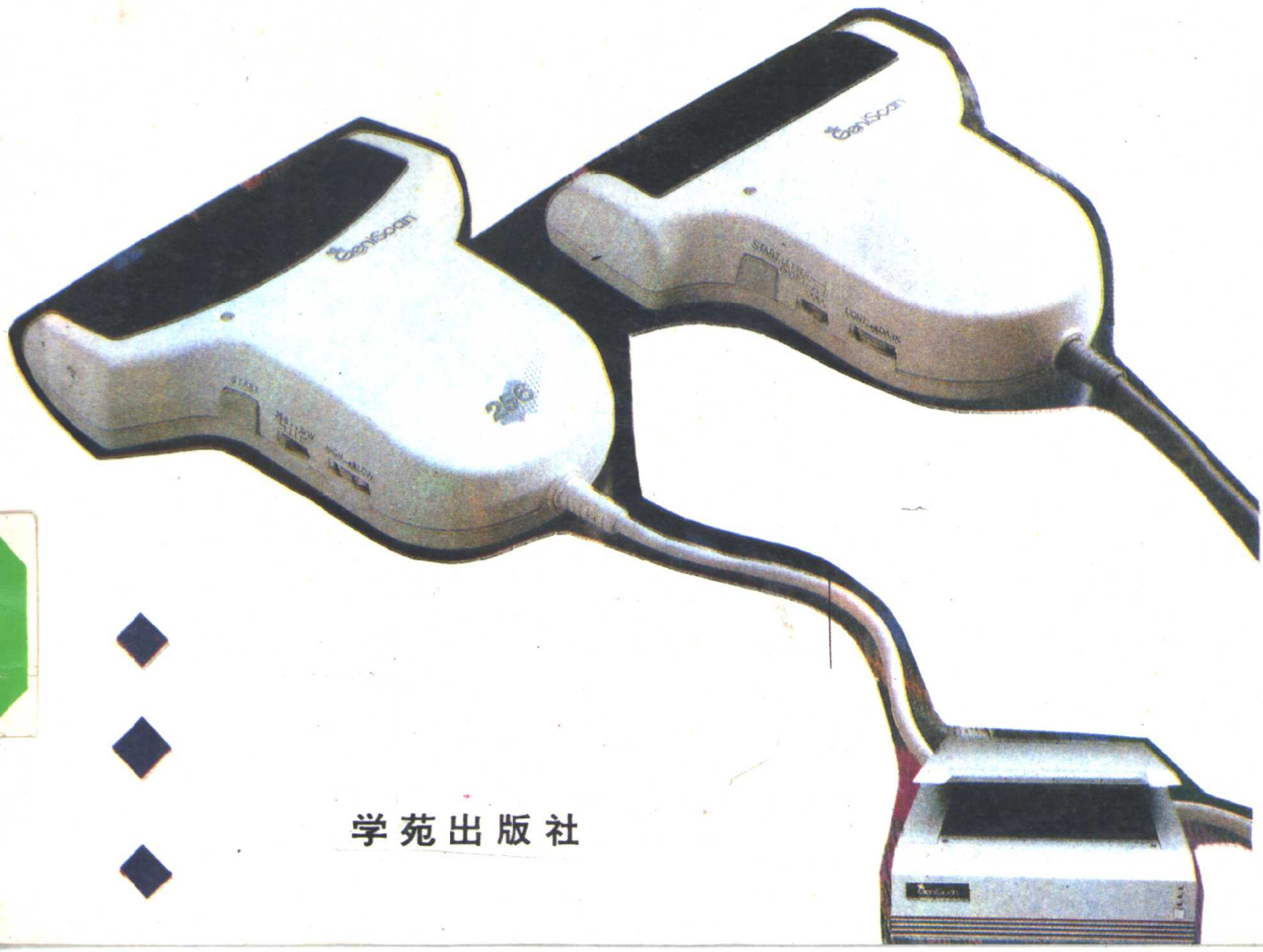
微机接口与应用系列丛书

高级 C++图形程序设计技术与应用

高利佳　主编



学苑出版社

微机接口与应用系列丛书

高级 C++图形程序设计技术与应用

高利佳　编写
肖　雨　审校

学苑出版社
1993.

(京)新登字151号

内 容 提 要

本书系统地讲述了四个方面的内容：普通编程技巧、二维图形应用程序、三维图形应用程序和动画。每一部分都给出了大量的应用程序示例。全书的内容主要包括：鼠标/键盘/磁盘编程、交互式菜单图形和桌面印刷图形程序设计、交互三维图形程序设计、动画系列的分类与设计。有关动画和三维图形的绝妙的理论分析和恰到好处的实际应用是本书的一大特点。本书还给出了图形编程过程中的几种错误处理方法。另外，附录中的类库源代码实际上为读者提供了一个完整的C++图形类库。本书中的程序可用 Turbo C++、Borland C++、Zortech C++、Microsoft C++编译、链接和运行。

微机接口与应用系列丛书

高级C++图形程序设计技术与应用

编　　写：高利佳
审　　校：肖　雨
责任编辑：甄国宪
出版发行：学苑出版社　　**邮政编码**：100032
社　　址：北京市西城区成方街33号
印　　刷：双青印刷厂
开　　本：787×1092　1/16
印　　张：27.375　　**字　数**：634千字
印　　数：1～3000册
版　　次：1993年12月北京第1版第1次
ISBN7－5077－0803－9/TP・14
本册定价：21.00元

目 录

第一部分　高性能编程技巧

第一章 用C++开发图形应用程序

九十年代初期出现了一种新的编程语言，随之出现了新的编译器。它可提高编程效率，并提供可重用代码。1990年出现了C++。Zortech C++逐步走向市场，并形成了一个系列。到1990年末，Turbo C++已经成为Zortech C++的竞争者。到1991年初，Turbo C++的超集Borland C++进入C软件市场。而Microsoft公司声称在1991年底向市场投放一种C++编译码。C++几乎在一夜之间成为市场主流。C语言作为最为流行的编程语言，受到了极为严重的威胁，不是吗？

C程序设计语言是七十年代中期由Dennis Ritchie在AT&T贝尔实验室开发的。C语言最初设计为一种系统编程语言。然而，它却迅速地成为各种软件开发者所选用的语言。C语言提供了许多如BASIC、Pascal以及Fortran等高级语言所能提供的性能特点。同时，它还可以同汇编语言等低级语言一样，直接同硬件打交道。

C++编程语言是八十年代初期由Bjarne Stroutrup在AT&T贝尔实验室开发的。C++是一种仿真语言，它依赖于面向对象编程(OOP)。Stroustrup最初将其作为一种解释器或者作为标准C编译器的处理程序。C++处理程序将C++源代码翻译成C源代码，以便C编译器在分析和编译时使用。在九十年代初期，可以接收C++源代码并产生目标代码的C++编译器开始进入市场。

相对C编程语言而言，C++在以下两个方面进行了改进。首先，C++对C进行了扩展，如新的关键字和函数等。其次，它增加了面向对象编程的高效性能。

然而，更为重要的是C++与C完全兼容。由于C++是C的超集，因此，可将C++编译器视为普通C编译器。这就意味着我们可以逐步由C过渡到C++编程。事实上，C++的确是C的一个超集。

1.1 C++对C的扩展

C++对C进行了非常有用的扩展。这些扩展可分为四个主要方面，即注释、变量、函数以及内存分配。

1.2 C++注释

C++支持C的注释标记。这种格式规定，字符串"/*"后的所有字符均被编译器忽略，直到遇到字符串"*/"为止。以下为C和C++源代码中的合法注释：

```
/* this is a comment */
```

C++还提供了另一种注释格式：双斜线"//"后的所有字符均被编译器忽略，直到这一行源代码结束为止。下面这一行为C++源代码中的合法注释：

```
//this is a comment
```

大多数C编译器都会认为这是语法错误。

双斜线注释格式使我们很容易在源代码中加入注释。由于注释行自动在当前行末尾结束，因此，不必考虑终止注释。如果想插入注释，只需延长一行，这要比/*…*/格式简单。

许多C编译器都不能识别双斜线注释格式。

1.3 C++变量

C++对C程序员在程序中处理变量的方法进行了改进。其中，两个最有用的扩展是变量声明的位置以及C++常量关键字。

变量声明的位置

C编程协议约定，函数内部的所有变量必须在执行语句之前声明。全程变量必须在函数外部声明，并且要在使用了这些变量的执行语句之前声明。

C++放宽了源代码中声明变量的位置和时间限制。首先，C++允许在程序中任何位置声明变量——当然要在使用此变量之前。其次，C++允许在第一次使用新变量的时候声明新的变量。下面的循环结构就是一例：

```
for (int Count = 1; Count < 10; Count++)
```

在这个例子中，变量Count在分配值1的同时被声明为整形。C++的这种变量声明特点使用户可以一直往下写程序，而不是不断在前面插入变量声明来满足后将面要写的程序的要求。

C++常数变量

常数变量是在程序的执行过程中其值不变的变量。如图1—1所示，C++提供更多的控

制，以便编译器处理这些常数。以关键字 const 开始的一行声明常数变量的类型，并为它赋值。

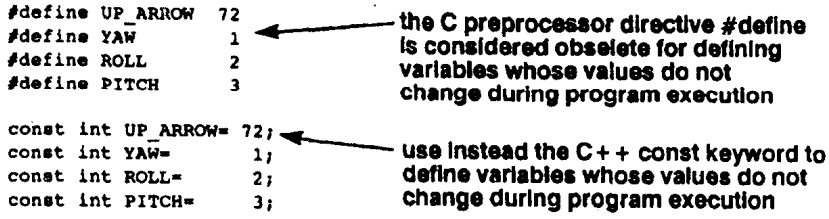


图 1—1　C++中的 const 关键字表示常数变量的值在程序执行过程中不变

由 const 常量类型说明符声明的任何变量均是只读变量，仅在其初始化时才能为其赋值。任何可以声明普通变量的地方均可声明并初始化 const 变量。然而，const 变量必须初始化为 27、70711、28044 等，不能将普通变量赋给 const 变量。

许多 C++程序员已经不使用 C 的 #define 语句了。

1.4 C++函数

C++对 C 程序员使用函数的方法进行了很有意义的扩展，其中两个重要的扩展是函数参数的隐含以及函数重载。

隐含函数参数

隐含函数参数是指调用函数时由编译器自动提供参数。如下例所示：

```
static int DrawDot (int,int,int Color = 4);
```

这行代码就是一个函数声明。声明的函数名为 DrawDot()，它返回一个整形值。DrawDot()函数声明为静态的，这就意味着它的作用域只限于声明它的源文件。DrawDot()函数在调用时需要三个参数，它们都是整形参数。如果在调用 DrawDot()函数时只提供了两个参数，编译器将会认为第三个参数为 4。如果在调用时提供三个参数，编译器将使用所提供的三个参数，而不使用隐含参数。如果在调用时提供三个参数，就意味着要用所提供的参数值覆盖隐含参数。

例如，假定函数 DrawDot()在显示屏幕上显示一个点。第一个参数可能是 x 坐标，第二个参数可能是 y 坐标，第三个参数可能是颜色。如果程序中以颜色 4 作为第三个参数来反复调用此函数，最好将 4 作为隐含参数。有时，如果需要调用 DrawDot()函数显示一个不以颜色 4 显示的点，则只需将要求的颜色作为第三个参数来调用此函数即可。否则，只需两个参数就可调用 DrawDot()函数。

函数可以声明为带有多个隐含参数，但隐含参数必须放在函数参数的最后。本章后面部分

的图 1－6 就是一个多隐含参数的例子。

注意，隐含参数必须放在函数的最后加以声明。

函数重载

C＋＋函数重载是指通过同一函数名调用不同的函数。下面的例子声明了三个不同的函数，每个函数在屏幕上画一条线。

```
static void DrawTheLine (int,int);
static void DrawTheLine (int,int,int);
static void DrawTheLine (int,int,int,int);
```

第一个 DrawTheLine()函数定义可能包括图形绘制语句，即画一条线。函数所需要的参数为 x 坐标和 y 坐标。线段可能是从当前位置画到点(x,y)处。线段可能是由当前系统颜色填充的。

第二个 DrawTheLine()函数定义要求在调用函数时给出三个参数。第一个参数可能是 x 坐标，第二个参数可能是 y 坐标，第三个参数可能是所要求的颜色。函数可能在当前位置与(x,y)坐标点间画一条所要求颜色的实心线。

第三个函数 DrawTheLine()定义要求在调用时给出四个参数。第四个参数可能用来指定直线的类型——实线、虚线、点划线、图案线等。

C＋＋编译器将产生代码，根据函数调用时提供的参数的数目和类型来自动选择合适的 DrawTheLine()函数。不同的函数定义(可执行代码)可以使用同一函数名，但其参数不同。下例仍是 DrawTheLine()函数中的一个：

```
static void DrawTheLine (float,float,float,float);
```

尽管此 DrawTheLine()函数包含四个参数，它仍然不同于前面所举的包含四个参数的例子，因为前面包含的是四个整型参数。本例的四个参数为浮点类型。假如要求函数画一个带有参数的曲线而不是一条简单的直线，也许会用到上述带有浮点类型参数的函数。

函数重载通过给那些只在形式上变化的函数提供了一个公共函数名，使得编程更加容易。本章后面的图 1－6 就是类方法中使用的函数重载的例子，也就是所谓的“多态性”。

1.5 C＋＋ new 和 delete 关键字

C＋＋提供两个新的关键字，使程序在执行过程中很容易分配和释放内存空间，如图 1－2 所示。

操作符 new 从空闲存储区分配内存。操作符 delete 将已分配的内存返回给空闲存储区。空

闲存储区是堆或远程堆，这取决于所使用的存储器模式。空闲存储区指空闲的内存空间。

C++ new 操作符

new 操作符用来为数据类型、结构或数组分配适当的内存块。此操作符将自动分配恰当的内存块大小。如图 1－2 所示，new 操作符产生易读的源代码，并能为减少许多低级烦琐的程序代码。new 操作符比 C 语言中的 malloc 更好用，它使源代码可读性强，并且易于维护。

C++ delete 操作符

delete 操作符释放由 new 操作符分配的内存空间。图 1－2 所示为其执行过程。

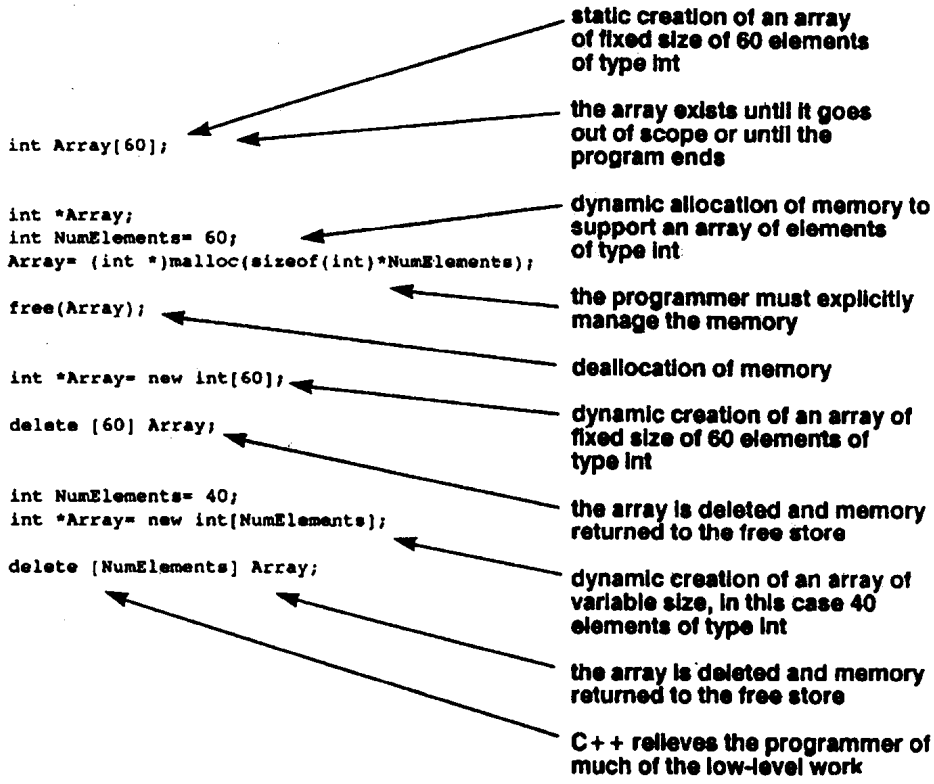


图 1－2　C++new 和 delete 操作符为程序执行提高强大的内存分配能力

动态数组分配

new 和 delete 操作符的最为重要的意义还在于它们提供了分配和释放动态数组的功能。动态数组是指程序运行时的数组，通常为可变维数而不是常量维数。这是对 C 语言的很有意义的改进，如图 1－2 所示。new 和 delete 操作符还可用于建立和取消 C++对象。C++对象是指类的实例。

1.6 C++类

除了上面所提到的C++对C语言的扩展之外，C++还使开发环境焕然一新——强有力的面向对象的程序设计。

面向对象程序设计的核心是对象这一概念。对象是指一种新的结构类型的实例，如变量就是数据类型的实例。这种新的结构类型具有独特的功能，它给编程带来了新的活力。

新的结构类型

C++所提供的这种新的结构类型称为类。C++中的类类似于C中的结构，但二者之间具有很大的区别。C结构是变量类型，它是数据元素的集合；而C++类是由数据和对数据进行操作的函数构成的。C++类的声明同C结构的声明类似，只是这种新的变量类型包含用以操作数据的函数。数据称为数据成员，函数称为成员函数。

C++类是由数据和操作数据的函数组成的变量结构。

当用C声明了一个结构类型之后，就可以建立许多这种类型的变量。同样，在C++中，当声明了类之后，也可以建立许多这种类型的变量。类中的每一个变量称为实例，或对象。

在程序设计中使用对象，才能充分利用封装、继承和多态性。

封装

封装是指一组数据和对数据进行操作的函数之间的联系。由于数据和函数都可以视为对象，因此，不必仔细考虑它们的区别。封装隐藏了许多复杂性。这就意味着可以在实体外面进行处理和设计——这是一种更为高级的处理。C++封装就如汽车中的自动传动装置——再也不必担心齿轮的变化，而可以将注意力集中于公路、交通信号灯和沿街路标。

继承

继承是指从已建立的类中建立新的类。这就意味着可以重用代码。例如，假定想在正确的类中增加一些新的特性，可以利用C++继承来派生一个新类，使这个类包括原类的所有特性。然后，就可简单地将新的代码加到新的类中。C++编译器处理包含在现存代码中的所有低级操作——用户只需注意新的改进。由于原来的类仍然存在，因此，在程序的其它地方仍可使用它，不会产生意想不到的副作用。

C++继承使我们很容易重用那些已经测试和调试过的代码。

从已存在的类中派生的新类称为派生类。已存在的类称为基类。派生类包含基类的所有